UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/007,593	12/05/2001	Roy F. Brabson	RSW920010174US1	3527

7590 08/24/2005	EXAMINER
Jerry W. Herndon	SHAW, YIN CHEN

ART UNIT	PAPER NUMBER
2135	

Jerry W. Herndon
IBM Corporation T81/503
PO Box 12195
Research Triangle Park, NC 27709

DATE MAILED: 08/24/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

	Application No.	Applicant(s)
Office Action Summary	10/007,593	BRABSON ET AL.
	Examiner	Art Unit	
	Yin-Chen Shaw	2135	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE <u>3</u> MONTH(S) FROM
THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
   Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1) ☒ Responsive to communication(s) filed on <u>05 December 2001</u>.

2a) ☐ This action is **FINAL**.   2b) ☒ This action is non-final.

3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4) ☒ Claim(s) <u>1-26</u> is/are pending in the application.

   4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5) ☐ Claim(s) _____ is/are allowed.

6) ☒ Claim(s) <u>1-26</u> is/are rejected.

7) ☐ Claim(s) _____ is/are objected to.

8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9) ☐ The specification is objected to by the Examiner.

10) ☒ The drawing(s) filed on <u>05 December 2001</u> is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.

   Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

   Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

   a) ☐ All   b) ☐ Some * c) ☐ None of:

   1. ☐ Certified copies of the priority documents have been received.

   2. ☐ Certified copies of the priority documents have been received in Application No. _____.

   3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

   * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
   Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413)
   Paper No(s)/Mail Date. _____.

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-26 have been submitted for examination.

2. Claims 1-26 have been examined and rejected.

Specification

3. The disclosure is objected to because of the following informalities:

   a. The term, ("TTP") in line 11 of [0045] appears to be a typographical error. The acronym for "File Transfer Protocol" is ("FTP"). Appropriate correction is required.

   b. The acronym, "TLS", is used for denoting different terms ("Transport Layer Security" in the abstract and "Transaction Layer Security" in the Background). The correct corresponding term for "TLS", as specified in the RFC 2246 and other technical literatures, is Transport Layer Security. Appropriate correction is required.

Claim Objections

4. Claim 23 is objected to because of the following informalities:

   a. The term, "Transaction Layer Security", should be "Transport Layer Security". For examining purpose, "Transaction Layer Security" is considered as "Transport Layer Security", and appropriate correction is required.

Claim Interpretation

5. Claims have been afforded their broadest reasonable interpretation. Applicant's

language directed to invoking is interpreted equivalent to calling through the API

(application program interface).

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all

obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 1-4, 21-22, and 24-26 are rejected under 35 U.S.C. 103(a) as being

unpatentable over Berg (U.S. Pub. 2002/0116605) and further in view of Berg et

al. (U.S. Pub. 2001/0044904).

    a. *Referring to Claim 1:*

        As per Claim 1, Berg (0116605) disclose a method of improving security

processing in a computing network, comprising steps of:

providing an application program which makes use of the operating

system kernel during execution **[i.e., The protocol stack includes a

series of routines for processing packets. Conventionally, the

protocol stack has been part of the OS and has executed in kernel**

mode (lines 8-9 in [0053]). In FIG. 13, a socket application includes instructions for initiating the formation of a socket by calling a system function (or by calling an application program interface ("API")) to form a socket of a specific type (e.g. UDP or TCP) within a socket layer. In response to instructions of the OS kernel, the main board circuitry manages the socket layer. In response to such a call, the OS kernel includes instructions for forming the socket and returning a file descriptor (which references the socket) to the application (lines 4-13 in [0182])];

executing the application program [i.e., execution of a software application (lines 2-3 in [0008])];

Berg does not expressly disclose the remaining limitations of the claim. However, Berg et al. (0044904) disclose

providing security processing in an operating system kernel [i.e., This use of KSOCKS 74 and module 29 advantageously enables communication to occur directly wit the kernel space 26 without the need for any user-space 24 intermediary on the destination system (lines 10-14 in [0026]). Module (i.e., KCMAPI) 29 preferably includes a kernel-level version of Configuration and Management API 60 (lines 1-3 in [0027]). KCMAPI 29 includes encryption module 37 which enables authentication of both the source and destination of these communications and encryption (lines 13-16 in [0027])];

selectably securing at least one communication and using the provided

security processing in the operating system kernel **[i.e., The method**

includes the step of locating an authentication module in the kernel

space, in communicably coupled relation with the kernel-level

components, to selectively encrypt and decrypt communications

between the kernel-level components and a remote site (lines 4-9 in

[0006]). This use of KSOCKS 74 and module 29 advantageously

enables communication to occur directly wit the kernel space 26

without the need for any user-space 24 intermediary on the

destination system (lines 10-14 in [0026]). Module (i.e., KCMAPI) 29

preferably includes a kernel-level version of Configuration and

Management API 60 (lines 1-3 in [0027]). KCMAPI 29 includes

encryption module 37 which enables authentication of both the

source and destination of these communications and encryption

(lines 13-16 in [0027])]. Berg and Berg et al. are from similar

technology relating to network security communications. It would have

been obvious to one of ordinary skill in the art at the time of invention

was made to combine Berg and Berg et al. to obtain the system which

can offload security processing to the kernel level while still being able to

provide the security processing to the executing application programs

since one would be motivated to avoid the application-level

administrative control functions of this application being suborned or

otherwise compromised, thereby providing unauthorized access (lines

21-23 in [0005] from Berg et al. (0044904)). Therefore, it would have

been obvious to combine Berg and Berg et al. obtain the invention as

specified in claim 1.

b. *Referring to Claim 2:*

As per Claim 2, Berg and Berg et al. discloses the method according to

claim 1. In addition, Berg (0116605) further disclose the step of

configuring one or more ports used by the provided application program

such that communications using the configured ports are to be secured;

and wherein the selectably securing step then secures all

communications using the configured ports **[i.e., Each IP packet port is

configurable to be full duplex and to accommodate a variety of port

protocols (e.g. Ethernet, ATM and FDDI). The synchronization port

is configurable in the same manner as an IP packet port or, in an

alternative embodiment, is specially configured. The configuration

of the synchronization port is selected according to a particular

application deployed on the server farm. With suitable circuitry,

extremely fast synchronization is achievable for a particular

application (lines 4-12 in [0096]). In an alternative embodiment, the

iNIC includes additional circuitry and firmware (for clarity not

shown in FIG. 3) for performing specified encryption operation

(lines 8-10 in [0097]). The protocol stack processor of server 2's**

iNIC (in response to instructions of its ipOS) adds suitable header information to the response packet and sends it to the client through the IP network-connected port (IP 123.123.123.3) of server 2's iNIC (lines 11-15 in [0103])].

c. *Referring to Claim 3:*

As per Claim 3, Berg and Berg et al. disclose the method according to claim 2. In addition, Berg et al. (0044904) disclose wherein the provided application program does not include code for security processing **[i.e., In a preferred embodiment, shown in phantom, transport module 27 includes a kernel-level communication API (also referred to as socket or KSOCKS) 74, which provides a kernel-based server (e.g., a TCP server) 76 and a conventional communication thread 78. Similarly, reverse operations are completed by CMPAI calls made in user-space 24 of system 11 and/or space 24' of remote system 13. In additional to these functions, KCMAPI 29 includes encryption module 37 which enables authentication of both the source and destination of these communications, and encryption (lines 10-16 in [0027]), *API call is required because the actual code is not within the application*].**

d. *Referring to Claim 4:*

As per Claim 4, Berg and Berg et al. disclose the method according to claim 2. In addition, Berg et al. disclose the configuring step further

comprises specifying information to be used by the selectably securing step **[i.e., As shown, configuration client 30 may include a suite of tools including a Configuration GUI 64 and a command line interface (CLIDE) 66. These are used for such operations as adding rules (discussed hereinbelow) and turning protection on and off. (lines 5-10 in [0043])]**.

e. *Referring to Claim 21:*

As per Claim 21, Berg and Berg et al. discloses the method according to claim 1. Berg and Berg et al. further disclose the provided security processing in the operating system kernel as in Claim 1. In addition, Berg et al. (0044904) disclose operates in a Transmission Control Protocol Layer **[i.e., One approach to provide such communication, especially given the possibility of remote configuration, is to use a well-known protocol such as TCP/IP. However, the Windows implementation of sockets (Winsocket) for TCP/IP communication is available only in user space. Opening socket for communication in the kernel is thus not inherently supported in Windows NT. To overcome this difficulty, system 10 of the present invention incorporates the aforementioned kernel socket 74 (lines 5-13 in [0064]). Transport module 27 includes a kernel-level communication API (also referred to as socket or KSOCKS) 74,**

which provides a kernel-based server (e.g., TCP server) 76 and a
conventional communication thread 78 (lines 2-5 in [0026])].

f. *Referring to Claim 22:*

As per Claim 22, Berg and Berg et al. discloses the method according to
claim 1. In addition, Berger (0116605) discloses wherein the provided
security processing implements Secure Sockets Layer [i.e., **Similarly,
the iNIC memory's process information includes an SSL (secure
socket layer) map table for mapping a specified SSL connection
(port 443) to one or more associated servers within the server farm
(lines 1-4 in [0206])]**.

g. *Referring to Claim 24:*

As per Claim 24, it encompasses limitations that are similar to those of
the method Claim 1. Berger et al. (0044904) further disclose in a
manner which is transparent [i.e., **Briefly described, calls to KCMAPO
implement driver routines and management functions to initial,
establish, and conduct communications within the kernel (lines 7-9
in [0027])**. If there is no violation, then the original NT™ system
services are called, all transparent to the user (lines 8-10 in [0152]);
*calling processing is performed through API, therefore, it is not
only transparent to the user, but also to the applications*] in addition
to a system for improving security processing in a computing network
[i.e., **a system is provided for securing communication between a**

remote site and kernel-level components of a computer having user

space and kernel space (lines 1-4 in [0008])].

h. _Referring to Claim 25:_

As per Claim 25, it encompasses limitations that are similar to those of

the method Claim 1. Therefore, it is rejected with the same rationale

applied against Claim 1 above. In additional, Berger et al. (0044904)

disclose a system for improving security processing in a computing

network **[i.e., a system is provided for securing communication**

between a remote site and kernel-level components of a computer

having user space and kernel space (lines 1-4 in [0008])].

i. _Referring to Claim 26:_

As per Claim 26, it encompasses limitations that are similar to those of

the method Claim 1. Therefore, it is rejected with the same rationale

applied against Claim 1 above. In additional, Berger et al. (0044904)

disclose a computer program product for improving security processing

in a computing network, the computer program product embodied on

one or more computer-readable media **[i.e., A further embodiment of**

the present invention includes an article of manufacture for

providing secure communications with kernel-level components of

a computer system having an operating system that includes user

space and kernel space. The article of manufacture includes a

computer usable medium having computer readable program code

embodied therein (lines 1-7 in [0009]). In a still further embodiment,

the present invention includes computer readable program code for

providing secure communications with kernel-level components of

a computer system having an operating system that includes user

space and kernel space (lines 1-5 in [0010]).

7. Claims 5-17, 19, and 23 are rejected under 35 U.S.C. 103(a) as being

unpatentable over Berg (U.S. Pub. 2002/0116605) and Berg et al. (U.S. Pub.

2001/0044904), and further in view of Mod_SSL manual (Apache mod_ssl

version 2.6).

    a. *Referring to Claim 5:*

        As per Claim 5, Berg and Berg et al. disclose the method according to

claim 4. Berg and Berg et al. do not expressly disclose wherein the

specified information comprises one or more of: authentication

information; cipher suites options; and security key input information.

However, Mod_SSL manual discloses a cipher suite is available for

negotiation in SSL handshake **[i.e., This complex directive uses a**

colon-separated cipher-spec string consisting of OpenSSL cipher

specifications to configure Cipher Suit the client is permitted to

negotiate in the SSL handshake phase (line 1-2, pg. 8 of Chapter

3)]. Berg, Berg et al., and Mod_SSL manual are from similar technology

relating to network security communications. It would have been

obvious to one of ordinary skill in the art at the time of invention was made to combine Berg, Berg et al., and Mod_SSL manual to allow the cipher suite be specified since one would be motivated to apply to the standard SSL handshake when a connection is established in per-server context (lines 3-4, pg. 8 of Chapter 3). Therefore, it would have been obvious to combine Berg, Berg et al., and Mod_SSL manual to obtain the invention as specified in claim 5.

b. *Referring to Claim 6:*

As per Claim 6, Berg and Berg et al. disclose the method according to claim 2. Berg and Berg et al. do not expressly disclose wherein the configuring step comprises one or more of: providing port definition statements; setting environment variables; and using job control language. However, Mod_SSL manual discloses environment variables are provided in the SSL module and can be backward compatible. **[i.e., this module provides a lot of SSL information as additional environment variable to the SSI and CGI namespace. For backward compatibility the information can be made available under different names, too (lines 30-32, pg. 20 of Chapter 3)].** Berg, Berg et al., and Mod_SSL manual are from similar technology relating to network security communications. It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine Berg, Berg et al., and Mod_SSL manual to have SSL directive configured through

the use of the environment variables since one would be motivated to

provides a lot of SSL information (line 30, pg. 20 of Chapter 3 from

Mod_SSL manual) and support backward compatibility to other SSL

solutions (line 5, pg. 1 of Chapter 4 from Mod_SSL manual). Therefore,

it would have been obvious to combine Berg, Berg et al., and Mod_SSL

manual to obtain the invention as specified in claim 6.

c. _Referring to Claim 7:_

As per Claim 7, Berg and Berg et al. disclose the method according to

claim 1. Berg and Berg et al. further disclose the step of providing, in the

secure processing as in Claim 1. Berg and Berg et al. do not expressly

disclose support for one or more security directives. However,

Mod_SSL manual discloses the different classes of directives used by

Mod_SSL **[i.e., Notice that there are three major classes of**

directives which are used by mod_ssl: First Global Directives (i.e.,

directives with context "server config"), which can occur inside the

server config files but only outside of any sectioning commands

like <VirualHost>. Second Per-server Directives (i.e., those with

context "server config, virtual host"), which can occur inside the

server config files both outside (for the main/default server) and

inside <VirtualHost> sections. And third Per-Directory Directives

(i.e., those with context "server config, virual host, directory,

.htaccess"), which can pretty much occur everywhere (lines 8-16,

pg.1 of Chapter 3)]. Berg, Berg et al., and Mod_SSL manual are from similar technology relating to network security communications. It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine Berg, Berg et al., and Mod_SSL manual to have various SSL directives being supported as the security processing for the application since one would be motivated to know how a particular mod_ssl functionality is actually configured or activated (lines 3-4, pg. 1 of Chapter 3 from Mod_SSL manual). Therefore, it would have been obvious to combine Berg, Berg et al., and Mod_SSL manual to obtain the invention as specified in claim 7.

d. _Referring to Claim 8:_

As per Claim 8, Berg, Berg et al., and Mod_SSL manual disclose the method according to claim 7. In addition, Berg (0116605) discloses the step of invoking, during execution of the provided application program **[i.e., FIG. 13 is a block diagram of the iNIC and main board circuitry of FIG. 3, according to the illustrative embodiments in which a socket application is related to a socket and its associated connection endpoint. In FIG. 13, a socket application includes instructions for initiating the formation of a socket by calling a system function (or by calling an application program interface ("API")) to form a socket of a specific type (e.g. UDP or TCP) within a socket layer. In response to instructions of the OS kernel, the**

main board circuitry manages the socket layer. In response to such a call, the OS kernel includes instructions for forming the socket and returning a file descriptor (which references the socket) to the application (lines 1-13 in [0182])], and Mod_SSL manual further discloses one or more security directives as in Claim 7.

e. *Referring to Claim 9:*

As per Claim 9, Berg, Berg et al., and Mod_SSL manual disclose the method according to claim 7. Mod_SSL manual further discloses wherein the provided security directives comprise one or more of:

access capability for a client certificate; access capability for a client identifier; a request to start operation of the selectably securing step; and a request to stop operation of the selectably securing step **[i.e., SSLEngine (line 16, pg. 6 of Chapter 3). This directive toggles the usage of the SSL/TLS Protocol Engine. This is usually used inside a <VirtualHost> section to enable SSL/TLS for a particular virtual host. By default the SSL/TLS Protocol Engine is disabled for both the main server and all configured virtual hosts (lines 26-28, pg. 6 of Chapter 3)]**.

f. *Referring to Claim 10:*

As per Claim 10, Berg, Berg et al., and Mod_SSL manual disclose the method according to claim 8. Berg and Berg et al. further disclose the step of invoking, the provided security directives, and the executing

application program as in Claim 8.   In addition, Mod_SSL manual

discloses the access capability, client certificate, and the step of

returning the client certification from the provided security processing in

response to the invocation **[i.e., SSLVerifyClient (line 1, pg. 14 of**

Chapter 3). This directive sets the Certificate verification level for

the Client Authentication. Notice that this directive can be used

both in per-server and per-directory context. In per-server context it

applies to the client authentication process used in the standard

SSL handshake when a connection is established. In per-directory

context it forces a SSL renegotation with the reconfigured client

verification level after the HTTP request was read but before the

HTTP response is sent (lines 11-15, pg. 14 of Chapter 3); *the*

process of authentication means that the certificate is required to

be accessed and then returned for verification].

g. *Referring to Claim 11:*

As per Claim 11, it encompasses limitations that are similar to those of

method Claims 10.   Therefore, it is rejected with the same rationale

applied against Claim 11 above.   In addition, Mod_SSL manual

discloses the client identification **[i.e., Certificate (line 11, pg. 14 of**

Chapter 3), *where the certificate would contain a distinguished*

names as defined by the X.509 standard, and the distinguished

name is used to provide an identity in a specific context (Table 2,

pg. 3-4 in Chapter 2)].

h. *Referring to Claim 12:*

As per Claim 12, the rejection of Claim 1 is incorporated. Claim 12

further encompasses limitations that are similar to those of method

Claims 7, 8, and 9. In addition, Mod_SSL manual discloses wherein the

selectably securing secures all communications **[i.e., SSLEngine on**

(line 19, pg. 6 of Chapter 3). This directive toggles the usage of the

SSL/TLS Protocol Engine. This is usually used inside a

<VirtualHost> section to enable SSL/TLS for a particular virtual

host. By default the SSL/TLS Protocol Engine is disabled for both

the main server and all configured virtual hosts (lines 26-28, pg. 6

of Chapter 3), *the security function, SSLEngine, would be applied*

to all the application or data when it is toggled on], and Berg

(0116605) discloses executing application program as in Claim 1.

i. *Referring to Claim 13:*

As per Claim 13, the rejection of Claim 1 is incorporated. Claim 13

further encompasses limitations that are similar to those of method

Claims 7, 8, and 9. In addition, Mod_SSL manual discloses wherein the

selectably securing step then stops securing all communications **[i.e.,**

SSLEngine off (line 19, pg. 6 of Chapter 3). This directive toggles

the usage of the SSL/TLS Protocol Engine. This is usually used

inside a <VirtualHost> section to enable SSL/TLS for a particular

virtual host. By default the SSL/TLS Protocol Engine is disabled for

both the main server and all configured virtual hosts (lines 26-28,

pg. 6 of Chapter 3), *the security function, SSLEngine, would no*

longer be applied to all the application or data when it is toggled

off], and Berg (0116605) discloses executing application program as in

Claim 1.

j. *Referring to Claim 14:*

As per Claim 14, the rejection of Claim 12 is incorporated. In addition,

Claim 14 encompasses limitations that are similar to those of the method

Claim 4. Therefore, it is rejected with the same rationale applied against

Claim 4 above.

k. *Referring to Claim 15:*

As per Claim 15, the rejection of Claim 14 is incorporated. In addition,

Claim 15 encompasses limitations that are similar to those of the method

Claim 5. Therefore, it is rejected with the same rationale applied against

Claim 5 above.

l. *Referring to Claim 16:*

As per Claim 16, Berg, Berg et al., and Mod_SSL manual disclose the

method according to claim 12.   Berg (0116605) further discloses a

decision to invoke and the executing application program **[i.e., The iNIC**

stores various tables of information in support of ipOS decisions

about packets and control of server farm resources. As shown in

FIG. 3, the tables include various information, such as state

information, routing information, process information, and protocol

stack information (lines 1-6 in [0093]). FIG. 13 is a block diagram of

the iNIC and main board circuitry of FIG. 3, according to the

illustrative embodiments in which a socket application is related to

a socket and its associated connection endpoint. In FIG. 13, a

socket application includes instructions for initiating the formation

of a socket by calling a system function (or by calling an

application program interface ("API")) to form a socket of a specific

type (e.g. UDP or TCP) within a socket layer. In response to

instructions of the OS kernel, the main board circuitry manages the

socket layer. In response to such a call, the OS kernel includes

instructions for forming the socket and returning a file descriptor

(which references the socket) to the application (lines 1-13 in

[0182])] in addition to the security directive disclosed in Claim 7.

m. *Referring to Claim 17:*

As per Claim 17, the rejection of Claim 12 is incorporated. Claim 17

encompasses limitations that are similar to those of the method Claim

16. Therefore, it is rejected with the same rationale applied against

Claim 16 above. In addition, Mod_SSL manual discloses a security

negotiation protocol [i.e., The protocol is designed to support a range

of choices for specific algorithms used for cryptography, digests, and signatures. This allows algorithm selection for specific servers to be made based on legal, export or other concerns, and also enables the protocol to take advantage of new algorithms. Choices are negotiated between client and server at the start of establishing a protocol session. (lines 38-42, pg. 5 of Chapter 2). SSLCipherSuite (line 29, pg. 7 of Chapter 3), where Cipher Suite available for negotiation in SSL handshake (line 30, pg. 7 of Chpater 3)].

n. *Referring to Claim 19:*

As per Claim 19, Berg and Berg et al. disclose the method according to claim 1. Berg and Berg et al. further disclose wherein the provided application program includes calls that invoke **[i.e. calling a system function (or by calling an application program interface ("API")) (lines 6-7 in [0182] from Berg (0116605))]**, the security processing and the corresponding security functions **[i.e., KCMAPI 29 includes encryption module 37 which enables authentication of both the source and destination of these communications and encryption (lines 13-16 in [0027] from Berg et al. (0044904)]**.

Berg and Berg et al. do not expressly disclose step of interpreting, in the provided security processing, the calls as being non-operative. However, the Mod_SSL manual discloses the SSLRequire directive,

which only allows the access when certain boolean condition is true **[i.e.,
SSLRequire (line 14, pg. 18 of Chapter 3) allow access only when
an arbitrarily complex boolean expression is true (line 15, pg. 18 of
Chapter 3);** *This means that under certain conditions (depending
on the setup of the expressions), the SSLRequire directive will be
treated as non-operative]*.

o. *Referring to Claim 23:*

As per Claim 23, Berg and Berg et al. discloses the method according to
claim 1. Berg and Berg et al. do not expressly disclose wherein the
provided security processing implements Transaction Layer Security.
However, Mod_SSL manual discloses the module can provide transport
layer security **[i.e., This module provides strong cryptography for the
Apache (v1.3) webserver via the Secure Socket Layer (SSL v2/v3)
and Transport Layer Security (TLS v1) protocols (lines 1-3, pg. 1 of
Chapter 1)]**. Berg, Berg et al., and Mod_SSL manual are from similar
technology relating to network security communications. It would have
been obvious to one of ordinary skill in the art at the time of invention
was made to combine Berg, Berg et al., and Mod_SSL manual to realize
the security process can either be implemented as SSL or TLS since
one would be motivated to establish the server environment such that
the clients can only connect with one of the provided protocols (lines 8-9,
pg. 7 of Chapter 3 from Mod_SSL manual). Therefore, it would have

been obvious to combine Berg, Berg et al., and Mod_SSL manual to

obtain the invention as specified in claim 23.

8. Claims 18 and 20 are rejected under 35 U.S.C. 103(a) as being unpatentable

over Berg (U.S. Pub. 2002/0116605) and Berg et al. (U.S. Pub. 2001/0044904),

and further in view of Golan (U.S. Patent 5,974,549).

   a. *Referring to Claim 18:*

      As per Claim 18, Berg and Berg et al. disclose the method according

      claim 1, Berg and Berg et al. further disclose wherein the provided

      application program includes calls that invoke **[i.e. calling a system**

      function (or by calling an application program interface ("API"))

      (lines 6-7 in [0182] from Berg (0116605))], the security processing and

      the corresponding security functions **[i.e., KCMAPI 29 includes**

      encryption module 37 which enables authentication of both the

      source and destination of these communications and encryption

      (lines 13-16 in [0027] from Berg et al. (0044904)].

      Berg. and Berg et al. do not expressly disclose the remaining limitations

      of the claim. However, Golan discloses the intercepting of the API calls

      made from the software **[i.e., Note that the method of the present**

      invention is applicable also to calls made to APIs that are not within

      the set of preselected APIs. The method is operative to intercept all

      calls made by the software component, i.e., API calls and non-API

calls. The API calls not in the preselected set must still be intercepted since they may call APIs that are in the preselected set later in the call chain (lines 66-67, Col. 14 and lines 1-4, Col. 15)]; and

executing, responsive to the interception, **[i.e., when the security monitor DLL intercepts and traps an attempt to load and execute a downloadable software component (lines 65-67, Col. 6)].**

Berg, Berg et al., and Golan are from similar technology relating to network security communications. It would have been obvious to one of ordinary skill in the art at the time of invention was made to combine Berg, Berg et al., and Golan to have the calls of application program be taken for performing the security processing since one would be motivated to have applications monitored then executed in a secure mode in which every software downloaded executes in a secure sandbox (lines 19-21, Col. 2 from Golan). Therefore, it would have been obvious to combine Berg, Berg et al., and Golan to obtain the invention as specified in claim 18.

b. _Referring to Claim 20:_

As per Claim 20, Berg, Berg et al., and Golan disclose the method according to claim 18. Berg, Berg et al., and Golan do not expressly disclose wherein the provided application program may be executed on a system which does not include the provided security processing. in the

operating system kernel, in which case the calls operate to perform

security processing instead of the selectably securing step. However,

Berg (0116605) discloses the security processing is performed by calling

the offloaded intelligent NIC **[i.e., The protocol stack includes a series**

of routines for processing packets. Conventionally, the protocol

stack has been part of the OS and has executed in kernel mode. By

comparison, in the illustrative embodiments, the iNIC's protocol

stack processor executes instructions to perform the protocol

stack operations. Accordingly, such operations are offloaded from

the OS (lines 1-7 in [0094]). Similarly, the iNIC memory's process

information includes an SSL (secure socket layer) map table for

mapping a specified SSL connection (port 443) to one or more

associated servers within the server farm (lines 1-4 in [0206])].

Therefore, It would have been obvious to one of ordinary skill in the art

at the time of invention was made to realize that no security processing

in the kernel is necessary when offloading such a process to the

intelligent NIC (thus, calls to request for performing security process is

needed) since one would be motivated to have the overall performance

and efficiency enhanced of deploying software applications with a server

farm through a global computer network (lines 4-6 in [0010] from Berg

(0116605)). Therefore, it would have been obvious to combine Berg,

Berg et al., and Golan to obtain the invention as specified in claim 20.

Conclusion

9. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

   a. Burstein (U.S. Pub. 2003/0079146) discloses the proxy program, which sends messages generated by the application program to the network in a way that is transparent to the application program. The proxy program calls functions of the firewall program (the security processing) via an application program interface (API). The firewall engine processes the packetized message according to the set of rules in the rule module. Based on the second new rule, the firewall engine sends the message through the TCP/IP stack, the network driver layer, the network adapter 160 and out to the network. The application program also generates a PORT command and sends it to the TCP/IP stack via the first socket. The PORT command includes the IP address and TCP port number that the remote computer is to use in contacting the application program.

   b. Winiger (U.S. Patent 5,845,068) discloses Kernel 44 mediates relationships 46 between processes of application 40 and selected resources 48, such as objects, services, and external application connecting to the processes of application 40. Kernel 44 includes a security process 50 ensuring that each process of application 40 communicates only with resources having a security classification consistent with a predetermined security policy. According to a

mandatory access control (MAC) system, for example, security process 50 ensures that processes 42a-42d only communicate with resources 48 at the same security classification as the corresponding process of application 40. In addition, Winiger discloses port identifier for a destination system comprises a port number specifying a particular resource, database, or service requested by the source application, and a sensitivity level. Once the port number and security label have been extracted, the kernel determines whether the requested port at the specified security level is in open status.

c. Kavsan (U.S. Patent 6,412,069) discloses the operating system which has an application space and a kernel space. The cryptographic service software performs cryptographic services in the kernel space of the operating system. The cryptographic service software includes a kernel space level application programming interface and cryptographic service module having a library of encryption algorithms. The cryptographic serice software allows one to write code at the driver level of the computer in a anner similar to the way the CrytpoAPI™ software does at the higher application level. Encryption algorithms may be used to encrypt signals at the driver level, such as at the Ethernet port or at the modem port.

d. Arrow et al. (U.S. Patent 6,226,751) disclose the security device securely transmits and receives data over the network independently and transparently, relying upon its own CPU to avoid depriving the host of processing bandwidth. The bandwidth may be needed to offload processing,

such as encryption. More importantly, however, independent transmission by the security device also prevents the host software from being able to bypass the security mechanisms. In addition, after configuration, initialization, and key exchange have taken place, secure communication between pairs of hosts is automatically and transparently managed by the respective security devices. User programs executing atop a security device equipped host only require access to a standard built-in networking application program interface (API) such as WinSock or TLI.

10. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Yin-Chen Shaw whose telephone number is 571-272-8593. The examiner can normally be reached on 8:15 to 4:15 M-F.
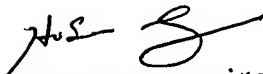
If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kim Yen Vu can be reached on 571-272-3859. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Primary Examiner
Art Unit 2135

YCS

Aug. 19, 2005